SVMs, Duality and the Kernel Trick

Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University
October 22\textsuperscript{nd}, 2007

Lagrange multipliers – Dual variables

Solving: \[
\min_x \max_\alpha \quad x^2 - \alpha (x - b)
\]
s.t. \( \alpha \geq 0 \)

\[
\frac{\partial L}{\partial x} = 2x - \alpha = 0 \Rightarrow x = 2x
\]
\[
\frac{\partial L}{\partial \alpha} = -(x - b)
\]

\( x = 1 \Rightarrow \frac{\partial L}{\partial x} = 0 \)
\( \alpha = 2 > 0 \)
Constraint relevant
Dual SVM formulation – the non-separable case

maximize \[ \sum \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j \]
\[ \sum_i \alpha_i y_i = 0 \]
\[ C \geq \alpha_i \geq 0 \]

\[ w = \sum_i \alpha_i y_i x_i \]
\[ b = y_k - w . x_k \]

for any \( k \) where \( C > \alpha_k > 0 \)

Why did we learn about the dual SVM?

- There are some quadratic programming algorithms that can solve the dual faster than the primal
- But, more importantly, the “kernel trick”!!!
  - Another little detour…
Reminder from last time: What if the data is not linearly separable?

Use features of features of features of features….

\[ \Phi(x) : \mathbb{R}^m \mapsto F \]

Feature space can get really large really quickly!

Higher order polynomials

\[ \text{num. terms} = \binom{d + m - 1}{d} = \frac{(d + m - 1)!}{d!(m - 1)!} \]

\[ d = 6, \ m = 100 \]

grows fast!

about 1.6 billion terms
Dual formulation only depends on dot-products, not on $w$!

$$\maximize_{\alpha} \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j$$
$$\sum_i \alpha_i y_i = 0$$
$$C \geq \alpha_i \geq 0$$

$$\maximize_{\alpha} \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$
$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$$
$$\sum_i \alpha_i y_i = 0$$
$$C \geq \alpha_i \geq 0$$

Dot-product of polynomials

$$\Phi(u) \cdot \Phi(v) = \text{polynomials of degree } d$$
Finally: the “kernel trick”!

\[
\text{maximize}_\alpha \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)
\]

\[
K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)
\]

\[
\sum_i \alpha_i y_i = 0
\]

\[
C \geq \alpha_i \geq 0
\]

- Never represent features explicitly
  - Compute dot products in closed form
- Constant-time high-dimensional dot-products for many classes of features
- Very interesting theory – Reproducing Kernel Hilbert Spaces
  - Not covered in detail in 10701/15781, more in 10702

Polynomial kernels

- All monomials of degree \(d\) in \(O(d)\) operations:
  \[
  \Phi(u) \cdot \Phi(v) = (u \cdot v)^d = \text{polynomials of degree } d
  \]

- How about all monomials of degree up to \(d\)?
  - Solution 0:
  - Better solution:
Common kernels

- Polynomials of degree $d$ $K(u, v) = (u \cdot v)^d$
- Polynomials of degree up to $d$ $K(u, v) = (u \cdot v + 1)^d$
- Gaussian kernels $K(u, v) = \exp \left(-\frac{||u - v||}{2\sigma^2}\right)$
- Sigmoid $K(u, v) = \tanh(\eta u \cdot v + \nu)$

Overfitting?

- Huge feature space with kernels, what about overfitting???
  - Maximizing margin leads to sparse set of support vectors
  - Some interesting theory says that SVMs search for simple hypothesis with large margin
  - Often robust to overfitting
What about at classification time

- For a new input $x$, if we need to represent $\Phi(x)$, we are in trouble!
- Recall classifier: $\text{sign}(w \cdot \Phi(x) + b)$
- Using kernels we are cool!

$$K(u, v) = \Phi(u) \cdot \Phi(v)$$

$$w = \sum_i \alpha_i y_i \Phi(x_i)$$

$$b = y_k - w \cdot \Phi(x_k)$$

for any $k$ where $C > \alpha_k > 0$

---

SVMs with kernels

- Choose a set of features and kernel function
- Solve dual problem to obtain support vectors $\alpha_i$
- At classification time, compute:

$$w \cdot \Phi(x) = \sum_i \alpha_i y_i K(x, x_i)$$

$$b = y_k - \sum_i \alpha_i y_i K(x_k, x_i)$$

for any $k$ where $C > \alpha_k > 0$

Classify as $\text{sign}(w \cdot \Phi(x) + b)$
Remember kernel regression?

1. \[ w_i = \exp(-D(x, \text{query})^2 / K)^2 \]
2. How to fit with the local points?
   Predict the weighted average of the outputs:
   predict = \( \frac{\sum w_i y_i}{\sum w_i} \)

SVMs v. Kernel Regression

**SVMs**

\[
\text{sign} \left( w \cdot \Phi(x) + b \right)
\]

or

\[
\text{sign} \left( \sum_i \alpha_i y_i K(x, x_i) + b \right)
\]

**Kernel Regression**

\[
\text{sign} \left( \frac{\sum_i y_i K(x, x_i)}{\sum_j K(x, x_j)} \right)
\]
SVMs v. Kernel Regression

\[ \text{SVMs} \quad \text{Kernel Regression} \]
\[ \text{sign} \left( w \cdot \Phi(x) + b \right) \quad \text{or} \quad \text{sign} \left( \frac{\sum y_i K(x, x_i)}{\sum K(x, x_i)} \right) \]

Differences:
- **SVMs:**
  - Learn weights \( \alpha_i \) (and bandwidth)
  - Often sparse solution
- **KR:**
  - Fixed “weights”, learn bandwidth
  - Solution may not be sparse
  - Much simpler to implement

What’s the difference between SVMs and Logistic Regression?

<table>
<thead>
<tr>
<th></th>
<th>SVMs</th>
<th>Logistic Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High dimensional</td>
<td></td>
<td></td>
</tr>
<tr>
<td>features with kernels</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

©2005-2007 Carlos Guestrin
Kernels in logistic regression

\[ P(Y = 1 \mid x, w) = \frac{1}{1 + e^{-(w \cdot \Phi(x) + b)}} \]

- Define weights in terms of support vectors:
  \[ w = \sum_{i} \alpha_i \Phi(x_i) \]
  \[ P(Y = 1 \mid x, w) = \frac{1}{1 + e^{-(\sum_i \alpha_i \Phi(x_i) \cdot \Phi(x) + b)}} = \frac{1}{1 + e^{-(\sum_i \alpha_i K(x,x_i) + b)}} \]

- Derive simple gradient descent rule on \( \alpha_i \)


What’s the difference between SVMs and Logistic Regression? (Revisited)

<table>
<thead>
<tr>
<th></th>
<th>SVMs</th>
<th>Logistic Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss function</td>
<td>Hinge loss</td>
<td>Log-loss</td>
</tr>
<tr>
<td>High dimensional features with kernels</td>
<td>Yes!</td>
<td>Yes!</td>
</tr>
</tbody>
</table>
What you need to know

- Dual SVM formulation
  - How it’s derived
- The kernel trick
- Derive polynomial kernel
- Common kernels
- Kernelized logistic regression
- Differences between SVMs and logistic regression

Announcements

- Midterm:
  - Thursday Oct. 25th, Thursday 5-6:30pm, MM A14
    - All content up to, and including SVMs and Kernels
      - Not learning theory
- Midterm review:
  - Tuesday, 5-6:30pm, location TBD
    - You should read midterms for Spring 2006 and 2007 before the review session
    - Then, you can ask about some of the questions in these midterms
What now…

- We have explored many ways of learning from data
- But…
  - How good is our classifier, really?
  - How much data do I need to make it “good enough”?
A simple setting…

- Classification
  - m data points
  - Finite number of possible hypothesis (e.g., dec. trees of depth d)
- A learner finds a hypothesis \( h \) that is consistent with training data
  - Gets zero error in training – \( \text{error}_{\text{train}}(h) = 0 \)
- What is the probability that \( h \) has more than \( \varepsilon \) true error?
  - \( \text{error}_{\text{true}}(h) \geq \varepsilon \)

How likely is a bad hypothesis to get \( m \) data points right?

- Hypothesis \( h \) that is consistent with training data \( \rightarrow \) got \( m \) i.i.d. points right
  - \( h \) “bad” if it gets all this data right, but has high true error
  - Prob. \( h \) with \( \text{error}_{\text{true}}(h) \geq \varepsilon \) gets one data point right

  - Prob. \( h \) with \( \text{error}_{\text{true}}(h) \geq \varepsilon \) gets \( m \) data points right
But there are many possible hypothesis that are consistent with training data

How likely is learner to pick a bad hypothesis

- Prob. $h$ with $\text{error}_\text{true}(h) \geq \varepsilon$ gets $m$ data points right

- There are $k$ hypothesis consistent with data
  - How likely is learner to pick a bad one?
Union bound

- $P(A \lor B \lor C \lor D \lor \ldots)$

How likely is learner to pick a bad hypothesis

- Prob. $h$ with $\text{error}_{\text{true}}(h) \geq \epsilon$ gets $m$ data points right

- There are $k$ hypothesis consistent with data
  - How likely is learner to pick a bad one?
Review: Generalization error in finite hypothesis spaces [Haussler ’88]

**Theorem:** Hypothesis space $H$ finite, dataset $D$ with $m$ i.i.d. samples, $0 < \epsilon < 1$ : for any learned hypothesis $h$ that is consistent on the training data:

$$P(\text{error}_{\text{true}}(h) > \epsilon) \leq |H| e^{-m\epsilon}$$

Using a PAC bound

Typically, 2 use cases:

- 1: Pick $\epsilon$ and $\delta$, give you $m$
- 2: Pick $m$ and $\delta$, give you $\epsilon$
Review: Generalization error in finite hypothesis spaces [Haussler ’88]

**Theorem**: Hypothesis space $H$ finite, dataset $D$ with $m$ i.i.d. samples, $0 < \epsilon < 1$ : for any learned hypothesis $h$ that is consistent on the training data:

$$P(\text{error}_{true}(h) > \epsilon) \leq |H|e^{-m\epsilon}$$

Even if $h$ makes zero errors in training data, may make errors in test

Limitations of Haussler ‘88 bound

- Consistent classifier
- Size of hypothesis space
What if our classifier does not have zero error on the training data?

- A learner with zero training errors may make mistakes in test set
- What about a learner with $\text{error}_{\text{train}}(h)$ in training set?

Simpler question: What’s the expected error of a hypothesis?

- The error of a hypothesis is like estimating the parameter of a coin!

- Chernoff bound: for $m$ i.i.d. coin flips, $x_1, \ldots, x_m$, where $x_i \in \{0,1\}$. For $0<\epsilon<1$:

$$P\left(\theta - \frac{1}{m} \sum_{i} x_i > \epsilon\right) \leq e^{-2m\epsilon^2}$$
Using Chernoff bound to estimate error of a single hypothesis

\[ P \left( \theta - \frac{1}{m} \sum_i x_i > \epsilon \right) \leq e^{-2m\epsilon^2} \]

But we are comparing many hypothesis: **Union bound**

For each hypothesis \( h_i \):

\[ P ( \text{error}_{true}(h_i) - \text{error}_{train}(h_i) > \epsilon ) \leq e^{-2m\epsilon^2} \]

What if I am comparing two hypothesis, \( h_1 \) and \( h_2 \)?
Generalization bound for $|H|$ hypothesis

- **Theorem:** Hypothesis space $H$ finite, dataset $D$ with $m$ i.i.d. samples, $0 < \varepsilon < 1$: for any learned hypothesis $h$:
  \[ P(\text{error}_{true}(h) - \text{error}_{train}(h) > \varepsilon) \leq |H|e^{-2m\varepsilon^2} \]

PAC bound and Bias-Variance tradeoff

- $P(\text{error}_{true}(h) - \text{error}_{train}(h) > \varepsilon) \leq |H|e^{-2m\varepsilon^2}$

  or, after moving some terms around, with probability at least $1-\delta$:
  \[
  \text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{\ln |H| + \frac{1}{2}\ln{\frac{1}{\delta}}}{2m}}
  \]

- **Important:** PAC bound holds for all $h$, but doesn’t guarantee that algorithm finds best $h$!!!
What about the size of the hypothesis space?

\[ m \geq \frac{1}{2\epsilon^2} \left( \ln |H| + \ln \frac{1}{\delta} \right) \]

- How large is the hypothesis space?

Boolean formulas with \( n \) binary features

\[ m \geq \frac{1}{2\epsilon^2} \left( \ln |H| + \ln \frac{1}{\delta} \right) \]
Number of decision trees of depth k

Recursive solution

Given n attributes

\( H_k = \text{Number of decision trees of depth } k \)

\( H_0 = 2 \)

\( H_{k+1} = (\text{#choices of root attribute}) \times (\text{# possible left subtrees}) \times (\text{# possible right subtrees}) \)

\( = n \times H_k \times H_k \)

Write \( L_k = \log_2 H_k \)

\( L_0 = 1 \)

\( L_{k+1} = \log_2 n + 2L_k \)

So \( L_k = (2^k-1)(1+\log_2 n) + 1 \)

---

PAC bound for decision trees of depth k

\[ m \geq \frac{1}{2\epsilon^2} \left( \ln |H| + \ln \frac{1}{\delta} \right) \]

- Bad!!!
  - Number of points is exponential in depth!

- But, for \( m \) data points, decision tree can’t get too big…

Number of leaves never more than number data points
Number of decision trees with k leaves

\[ H_k = \text{Number of decision trees with k leaves} \]
\[ H_0 = 2 \]
\[ H_{k+1} = n \sum_{i=1}^{k} H_i H_{k+1-i} \]

Loose bound:
\[ H_k = n^{k-1}(k+1)2^{k-1} \]

Reminder:
\[ |\text{DTs depth } k| = 2 \times (2n)^{2^{k-1}} \]

PAC bound for decision trees with k leaves – Bias-Variance revisited

\[ H_k = n^{k-1}(k+1)2^{k-1} \]
\[ \text{error}_\text{true}(h) \leq \text{error}_\text{train}(h) + \frac{\ln |f| + \ln \frac{1}{\delta}}{2m} \]
\[ \text{error}_\text{true}(h) \leq \text{error}_\text{train}(h) + \frac{(k-1) \ln n + (2k-1) \ln (k+1) + \ln \frac{1}{\delta}}{2m} \]
What did we learn from decision trees?

- Bias-Variance tradeoff formalized

$$\text{error}_{\text{true}}(h) \leq \text{error}_{\text{train}}(h) + \sqrt{\frac{(k - 1) \ln n + (2k - 1) \ln(k + 1) + \ln \frac{3}{\delta}}{2m}}$$

- Moral of the story:
  Complexity of learning not measured in terms of size hypothesis space, but in maximum number of points that allows consistent classification
  - Complexity $m$ – no bias, lots of variance
  - Lower than $m$ – some bias, less variance